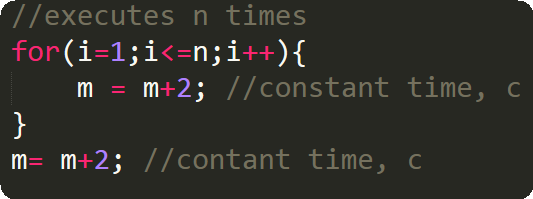
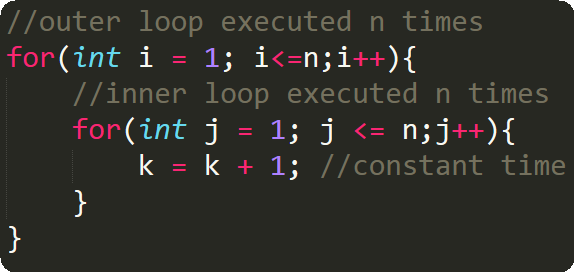
**Guidelines for Asymptotic Analysis**

**Loops: The running time of a loop is, at most, the running time of the statements inside the loop (including tests) multiplied by the number of iterations.**

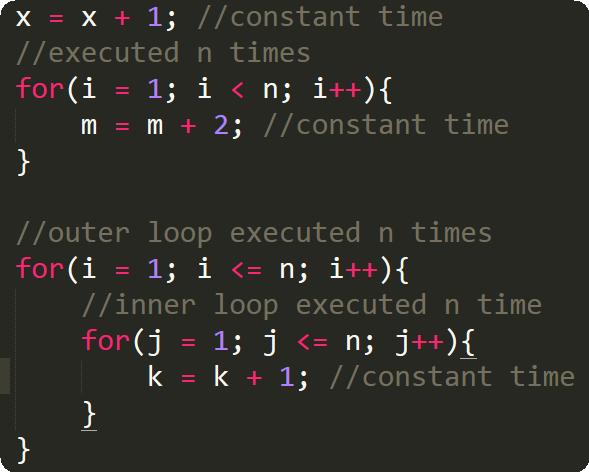


Total time = a constant c \* n = c n=O(n).

**Nested loops:** Analyze from inside out. Total amount time is the product of the sizes of all the loops.

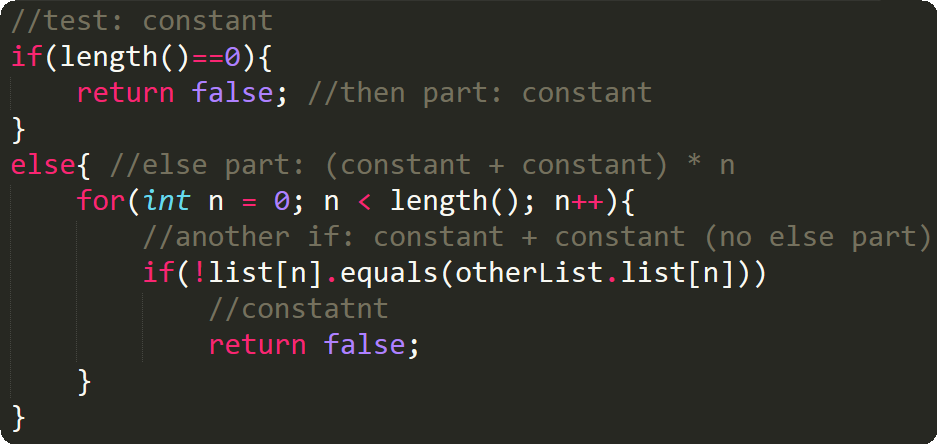


**Consecutive Statements:** Add the time complexities of each statement.

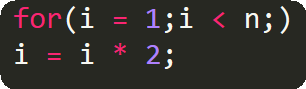


Total time = c0  + c1n + c2n2 ­= O(n­2)

**If-then-else statements:** ­­­­­­­Worst-case running time: the test, plus either the then part or the else part (whichever is the larger).



**Logarithmic complexity:** An algorithm is O(logn) if it takes a constant time to cut the problem size by a fraction (usually by ½). As an example let us consider the following program:



If we observe carefully, the value of  *i* is doubling every time. Initially *i = 1,* in the next step *I = 2,* and in subsequent steps *i =* 4, 8 and so on. Let us assume that the loop is executing some *k* times.

At k­th step 2k = *n* and we come out of loop. Taking logarithm on both sides, gives

*log(2k) = logn*

*klog2 = logn*

*k = logn //if we assume base-2*

*Total time = O(logN).*

*Note: Similarly, for the case below also, worst case rate of growth is O(logn). The same discussion holds good for decreasing sequence as well.*